Monitoring Environmental Noise and Its Relationship to Airborne Particulates

David R. Brooks, February, 2022. © 2022

Environmental justice issues often focus on air and water quality within minority and other underserved communities, arising from measurable sources of pollution. Noise is also part of the environmental landscape. It can be soothing, like the calling of birds or the rustle of a gentle breeze, or disturbing, like the crash of thunder, the roar of a tornado, or intrusive human activity.

In some circumstances, noise from human activities can be considered just by itself as a form of environmental pollution – it can disrupt other activities and it can cause stress, anxiety, hearing loss, and other health problems. (See https://en.wikipedia.org/wiki/Health_effects_from_noise for an introduction to this topic.)

Noise from some human activities may be directly associated with other measurable forms of pollution. For example, construction and traffic can increase the concentration of airborne particulates. Can these activities be associated with increased noise levels? Or to put it another way, is there a quantifiable relationship between noise levels and particulate emissions from these activities?

IESRE has developed airborne particulate monitors using Arduinos and laser-based particulate sensors. One version, with an LCD display suitable for indoor use or in a location well-protected from precipitation, is shown in Figure 1. The PM sensor is the blue unit on the right. Data are logged on the SD card. The PM values are for PM1, PM2.5, and PM10; these values – 2, 3, 4 – are small because the system is indoors in a clean office. The temperature sensor is the white unit to the right of the power cord. Temperature is in degrees Celsius and relative humidity is in percent.

Figure 1. PM sensor with T/RH sensor and LCD data display.

For the purposes of the Air Quality Index (AQI), particulates are reported as PM2.5 for particulates with approximate diameters less than or equal to 2.5 microns, and PM10 for particulates with approximate diameters less than or equal to 10 microns, both in units of $\mu g/m^3$. (PM2.5 includes the contributions from PM1 and smaller particles and PM10 includes the contributions from all particles smaller than 10 microns.)

Airborne particulates in densely populated urban communities, where minority and other underserved populations tend to be concentrated, are of special concern. Of course, there are many sources of airborne particulates in other settings as well, such as wind-blown dust from agricultural activity in rural areas. But of particular interest is the fact that airborne particulates in urban settings can be highly variable in both space and time. It's interesting to consider whether noise levels from construction and traffic can be associated with localized variations in particulate levels that may pose environmental hazards even though the associated AQI values, particulate counts averaged over 24 hours, may be much lower. (See this 2018 EPA document https://www.airnow.gov/sites/default/files/2020-05/aqi-technical-assistance-document-sept2018.pdf for how AQI values are calculated.)

For sound processing with Arduinos, there are many inexpensive electret mics, like the $7 BOB-12758 from sparkfun.com or ID-1063 from adafruit.com, shown in the left-hand image of Figure 2; for a size reference, the three solder connection points are on 0.1" centers. Used by themselves, these inexpensive analog-output devices are useful primarily for detecting and responding to sound events rather than providing a quantitative record of noise levels.

Figure 2. Electret mic module from adafruit.com and electret-mic-based sound level meter module from dfrobot.com.

Incorporating these electret mics into a calibrated sound level meter is a more complicated undertaking. Quantified sound levels are expressed in units of decibels, with dBA being the unit used to express noise levels adjusted to correspond to the frequency response distribution of "normal" human ears. The $40 SEN0232 Analog Sound Level Meter board from DFRobot.com, shown on the right-hand image, claims to serve as a calibrated dBA sound level meter when connected to a microcontroller to read and process its analog output (see https://www.dfrobot.com/product-1663.html); for a size reference, its electret microphone, centered at the left-hand edge of the board as shown here, is 0.5 cm in diameter. A three-wire cable is included (power, ground, analog out), requiring three M/M jumper or AWG22 solid wires to connect to Arduino headers or a breadboard.

Figure 3 shows DFRobot's board with an UNO and data logging shield that provides a real time clock for a date/time stamp on logged data and an SD card to store the data. This board will work with a Nano (for which data logging shields are also available) and it will also work with a 5 V Pro Mini with separate clock and SD card modules. With 5 V boards like the UNO, Nano, or Pro Mini, it's best to power the SEN0232 from the 3.3 V power pin because this is a predictable and well-regulated low-noise supply regardless of the voltage powering the board.

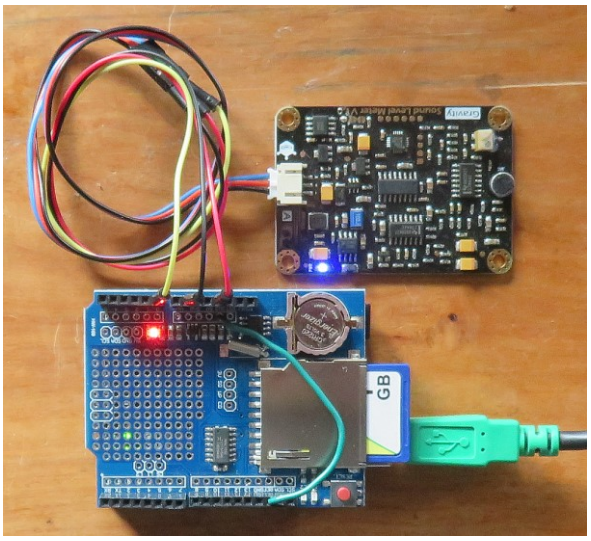


Figure 3. DFRobot sound level meter with UNO and data logging shield.

The code for this project also uses the 3.3 V power output pin on the UNO (or Nano, etc.) as an external reference voltage for processing the incoming analog voltage values – see the green wire soldered at the 3.3 V pin and attached to the AREF header pin. This gives slightly better A/D resolution for analog voltage inputs that don't exceed 3.3 V and a more dependable A/D/A process because the 3.3 V is independent of the voltage source powering the system. Although the 3.3 V power pin should be very close to 3.30 V,

it's best to actually measure it and use that value as the external reference voltage. For the system shown in Figure 3, the 3.3 V pin provided 3.27 V.

My (limited) experience with WiFi boards like the ESP32 indicate that they are not well suited for accurately processing analog inputs, although that problem can be solved by using an I2C ADS1115 programmable gain 16-bit A/D board available from adafruit.com and other online sources.

Here's the code for this project.

```
/* SoundMeterLog.ino, D. Brooks, February 2022
  See SoundMeterDFRobot.ino for the basic sketch for using this device3.
  DFRobot Gravity: Analog Sound Level Meter (SEN0232)
<https://www.dfrobot.com/wiki/index.php/Gravity:_Analog_Sound_Level_Meter_SKU:SEN0232>
  In order to protect the microphone on the board, do not touch the black
  membrane. Also you should keep it clean.
  Do not place the module on a conducting or semiconducting surface,
  as this could cause the microphone pins to be shorted.
  Default voltage on AREF pin is 5 V or 3.3 V if using a 3.3 V board.
  On 5 V boards that support an external reference voltage (UNO, Nano, etc.)
  it's a good idea to use the 3.3 V power pin as an external reference, as it's
  a low-noise and stable voltage source regardless of the board's power source.
  But in any case for most accurate results, measure the AREF value.
*/
// Data logging setup.
#include <Wire.h>
#include <SD.h>
#include <SPI.h>
#include <RTClib.h>
RTC_DS1307 rtc; // Assumes date/time has been set.
const int SDpin=10; // for UNO, Nano, etc.
File logfile;
char filename[]="DB_METER.CSV";
int YR,MON,DAY,HR,MIN,SEC;
// Data setup.
#define N 110 // # of 500 ms samples to average (average over ~1 min.
#define DELAYTIME 500
#define SoundSensorPin A0 // Input pin for board output.
#define VREF 3.27 // As measured at 3.3 V pin with system under power.
float voltageValue,dbValue;
float DbValue,dBAverage=0.,DbMin=1000.,DbMax=0.;
int i;
void setup() {
  Serial.begin(9600);
  Wire.begin(); rtc.begin();
  analogReference(EXTERNAL); // REQUIRED for using external reference!
  if (!SD.begin(SDpin)) {Serial.println("Card failed."); }
  Serial.println("card initialized.");
  logfile = SD.open(filename, FILE_WRITE);
  if (!logfile) {Serial.println("Couldn't create file."); }
  Serial.println("Starting...");
}
void loop() {
  for (i=1; i<=N; i++) {
    DbValue=analogRead(SoundSensorPin)/1023.*VREF;
    dBAverage+=DbValue;
    if(DbValue>DbMax)DbMax=DbValue;
    if(DbValue<DbMin) DbMin=DbValue;
```

```
    delay(DELAYTIME);
  }
  dBAverage/=N; dBAverage*=50.; //Convert voltages to decibel values.
  DbMin*=50.; DbMax*=50.;
  DateTime now=rtc.now();
  YR=now.year(); MON=now.month(); DAY=now.day();
  HR=now.hour(); MIN=now.minute(); SEC=now.second();
  // COM port output optional - can be commented out.
  Serial.print(YR); Serial.print('/'); Serial.print(MON); Serial.print('/');
  Serial.print(DAY); Serial.print(','); Serial.print(HR); Serial.print(':');
  Serial.print(MIN); Serial.print(':'); Serial.println(SEC);
  Serial.print("dBA_avg, dBA_min, dBA_max ");
  Serial.print(dBAverage,1); Serial.print(' ');
  Serial.print(DbMin,1); Serial.print(' ');Serial.println(DbMax,1);
  Serial.println("----------------");
  logfile.print(YR);logfile.print(',');logfile.print(MON);logfile.print(',');
  logfile.print(DAY);logfile.print(',');logfile.print(HR); logfile.print(',');
  logfile.print(MIN);logfile.print(',');logfile.print(SEC);
  logfile.print(',');
  logfile.print(DAY+HR/24.+MIN/1440.+SEC/86400.,5); logfile.print(',');
  logfile.print(dBAverage,1); logfile.print(',');
  logfile.print(DbMin,1); logfile.print(',');logfile.println(DbMax,1);
  logfile.flush();
  // Reset Db values.
  dBAverage=0.; DbMin=1000.; DbMax=0.;
}
```

Here's some data from running this code displayed in the Arduino IDE's COM port window, logging at roughly one-minute intervals; you can change the time interval by changing `DELAYTIME` or the number of samples `N`. The system is sitting on my desk, monitoring normal typing and paper-shuffling activities. Around 26 dBA is the baseline value for a very quiet environment. The maximum values are reasonable for this setting. However, checking the calibration for this system requires access to an independent calibrated dBA meter. For many purposes, absolute accuracy is not as important as consistent behavior that allows comparing data over time.

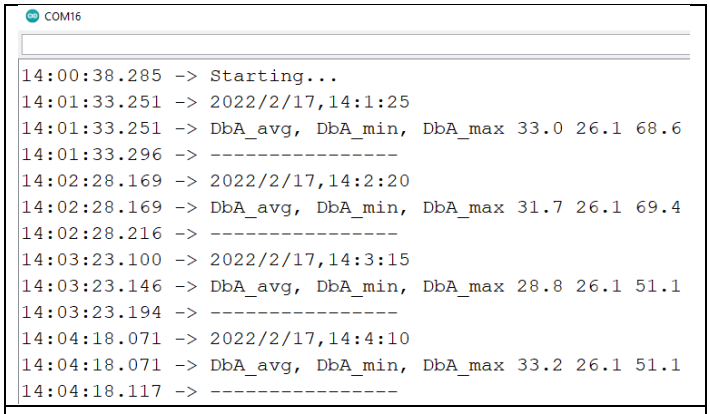


Figure 4. Some data generated by the project code.

Figure 5 shows some dBA values for various noise sources. Figure 6 shows some data recorded indoors in a house that faces a street with occasional traffic. This is a very quiet house with hot water radiator heat so there is no noise from fan-driven air. All the windows have storm windows that further insulate the inside of the house from outdoor noise. The spikes in the minimum value around day 9.94 (~10:30 pm) in Figure 6(a) indicate some activity in this part of the house, shown in more detail in 6(b).

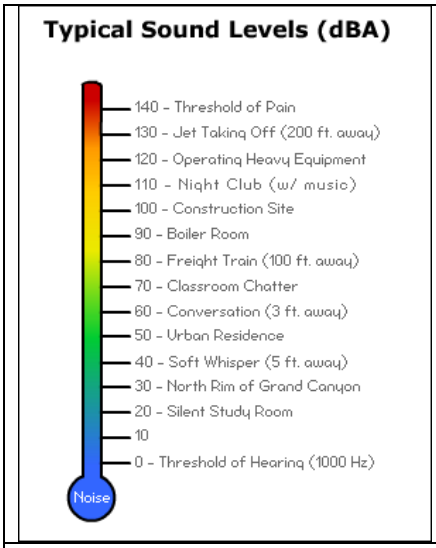


Figure 5. dBA values for various sound sources. (https://www.nexflow.com/blog/what-does-dBA-mean/)

Although the spikes in the minimum dBA values in Figure 6(b) look very large, this can be misleading. dBA values are a measure of sound intensity or power – a physically defined measurable quantity. An increase of 3 dBA corresponds to a doubling in sound power, but an increase or decrease of 10 dBA is required to produce a human-perceived doubling or halving of perceived loudness. Thus, the largest noise spike shown in Figure 6(b) represents only a doubling of perceived loudness relative to a very quiet background environment – even less than ongoing normal conversation as noted in Figure 5.

Figure 6(c) shows the average dBA values during the logging interval. The black line represents a 10-point trailing average of these values. It's believed that the occasional small "bumps," as first evident just before and after day 19.7, represent cars or trucks passing by in front of the house. These sounds are discernible if you're listening for them, but they aren't loud enough to be annoying or even distracting.

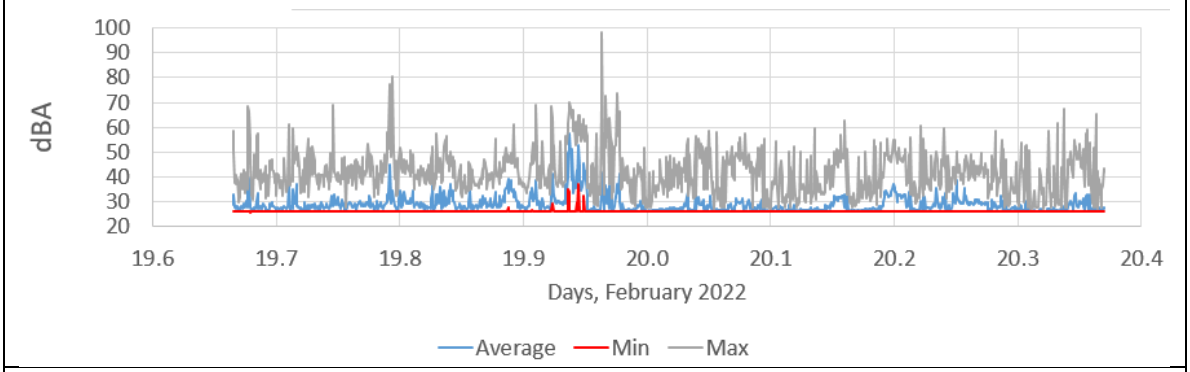
Persistent exposure to dBA values as low as 70 can cause permanent hearing damage, but values this high averaged over about one minute are never seen in this environment and only rarely seen as brief maximum values as shown in Figure 6(a).

Figure 6(d) shows another possible way to interpret these data. It shows the ratio of the maximum dBA to the average dBA values during the logging interval, again with a 10-point trailing average in black. A high value of the ratio indicates that the number of large dBA values is small compared to the total number of values during the logging interval – that is, some equivalent of a brief "spike" in sound rather than a generally louder environment. A ratio of 1.0, where the maximum values is the same as the average value, would mean that the noise environment was overall constant during the logging interval.

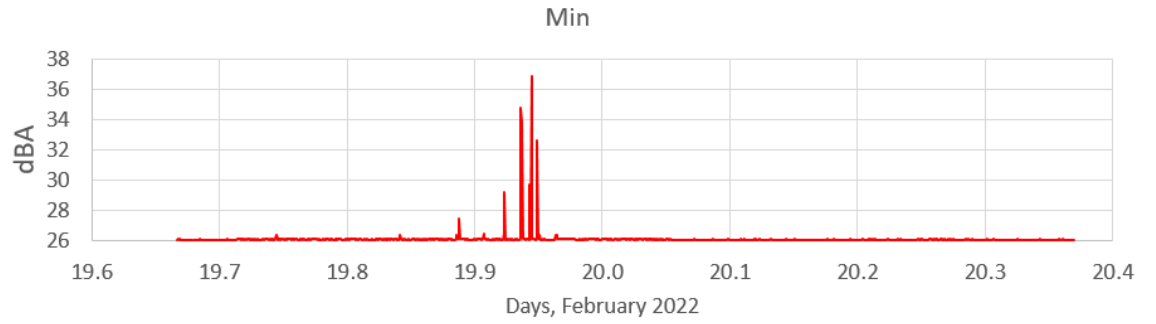
Figure 7 shows the same data presentations as Figure 6, but collected outdoors on a screened porch and depicting a much different sound environment. The minimum outdoor dBA values are much higher than inside a quiet house. It's harder to isolate specific activities, but the spike in the minimum dBA values between 23.5 and 23.6 (around 2:00 in the afternoon) is due to chainsaw activity several hundred feet away. Throughout the day there is occasional traffic on a road about 100 feet away. There may be some persistent wind noise, but everything quiets down during the night. In the morning, there are relatively long stretches of time during which the average noise is above 70 dBA. Many municipal noise ordinances prohibit persistent noise levels above 65 dBA, something like this:

"No person shall create or allow the creation of ongoing or constant noise such that the total ongoing or constant sound level audible outdoors, including the ambient sound level, exceeds 65

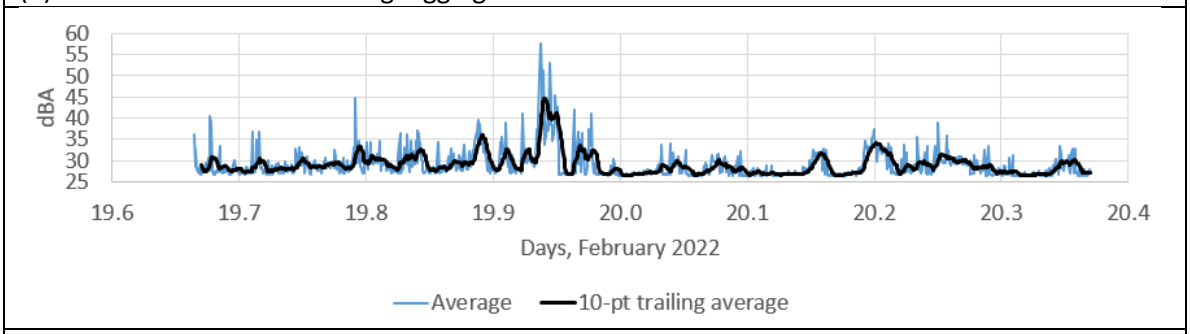
dB(A) during weekday or weekend daytime hours and 55 db(A) during weekday or weekend nighttime hours when measured at or outside any real property line."



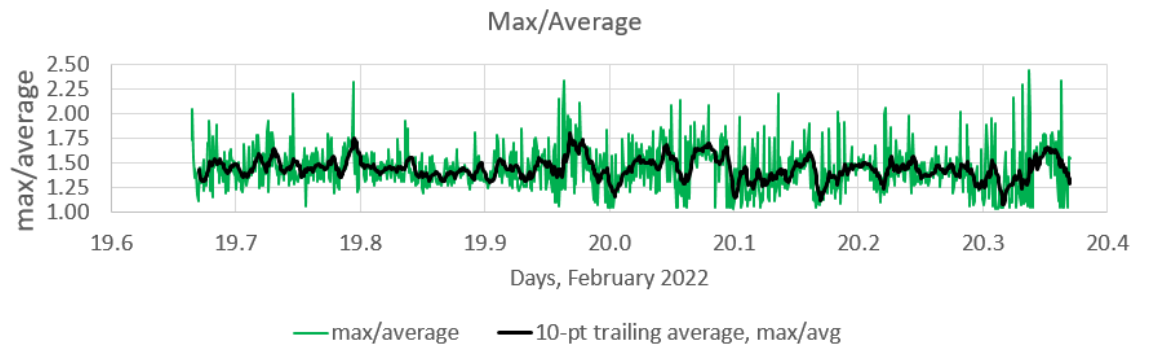
(a) 110 sound samples at 500 ms intervals.



(b) minimum dBA values during logging interval.

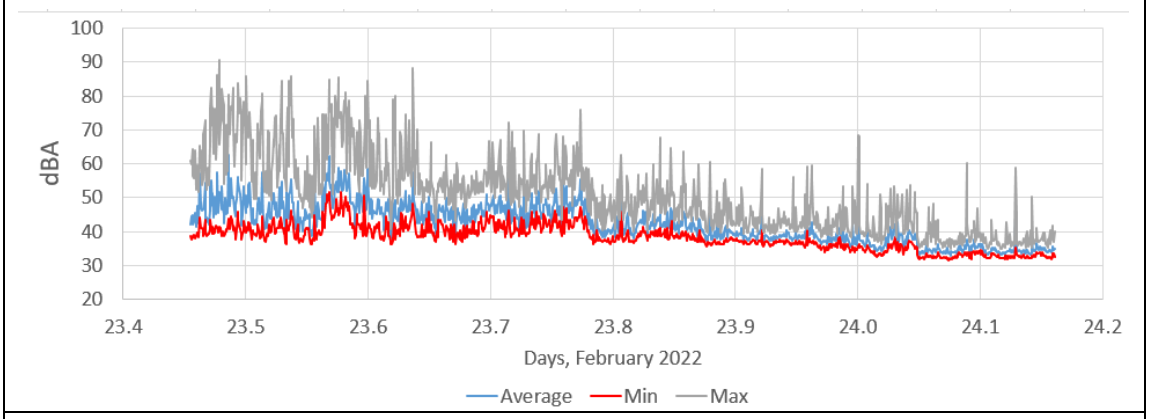


(c) average dBA values over the logging interval, with ~10 minute trailing average.

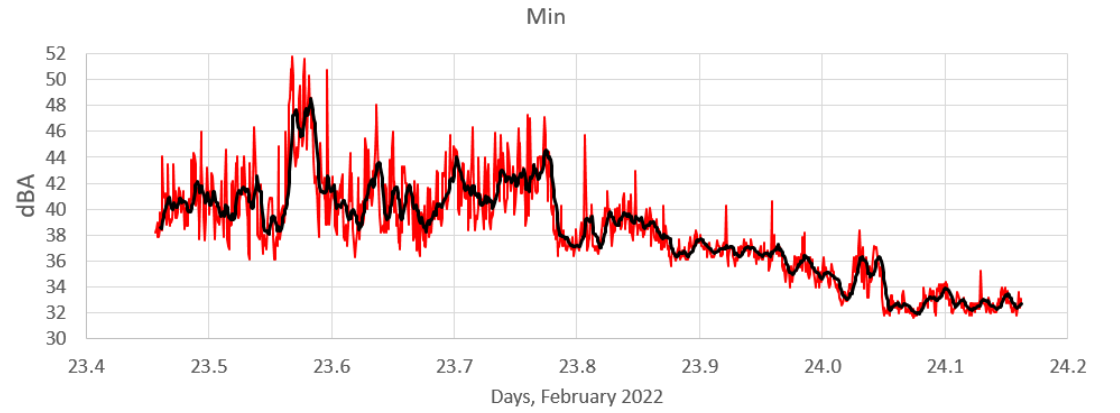


(d) Ratio of maximum to average logged dBA values with ~10 minute trailing average.

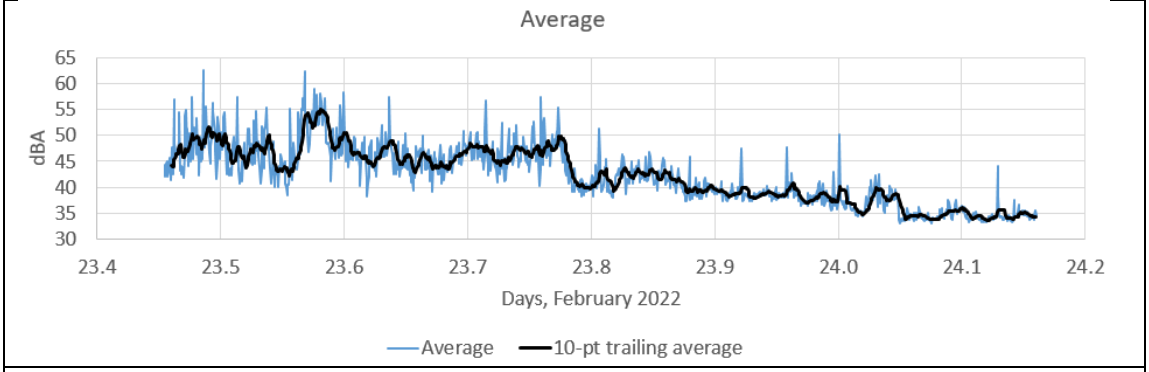
Figure 6. Data recorded inside a house.



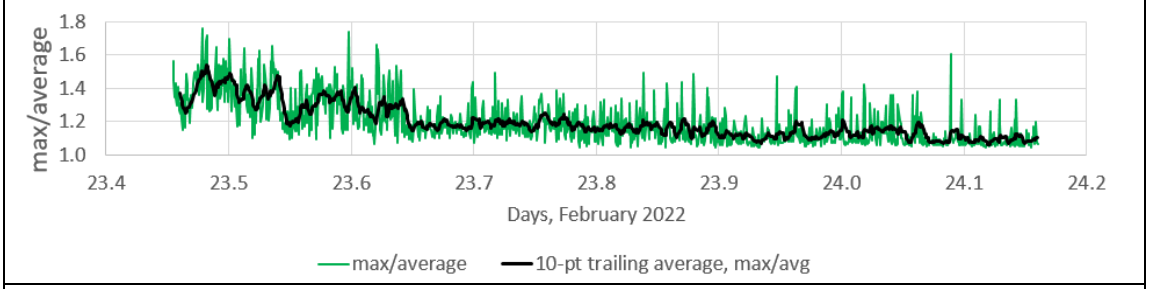
(a) 110 sound samples at 500 ms intervals.



(b) minimum dBA values during logging interval with ~10 minute trailing average.



(c) average dBA values over the logging interval with ~10 minute trailing average.



(d) Ratio of maximum to average logged dBA values with ~10 minute trailing average.

Figure 7. Data recorded outdoors on a screened porch.

The data shown in Figures 6 and 7 demonstrate that it should, in fact, be possible to correlate environmental noise levels with other data such as airborne particulate concentrations. The maximum levels even in Figure 7 are well below what can be experienced in urban environments or around construction sites anywhere, so there may be more pronounced data peaks and valleys in situations where higher intermittent or persistent localized noise levels are encountered..

The nature of this measurement, requiring an exposed microphone, does place some limitations on when and where this kind of system can be deployed. The primary constraint is that the microphone and the board to which it's attached must ***never*** be exposed to moisture of any kind. Wind can distort the sound environment as perceived by an electret microphone. It's important to keep the microphone surface clean – gently brushing away dust and dirt with a soft brush or paper tissue without actually touching the delicate surface. (It principle, it ***might*** be possible to replace a damaged microphone on the board because it's just attached with two solder points. Whether a replacement for a damaged microphone is available is a question to which I don't have an answer; unless it were an exact replacement, a new mic would in any case require recalibration of the unit.)

Because this system requires very little power, it should be possible to use batteries for outdoor projects – for example, 6 C or D cells in series or a small solar/battery system with a solar power manager board (for example, https://www.dfrobot.com/product-1712.html). 3.3 V Arduino boards with external data logging modules, like 3.3 V Pro Minis, are also possibilities as long as those boards allow reliable processing for analog inputs (to the best of my knowledge, ESP32 and perhaps ESP8266 WiFi project development boards won't do that without using separate I2C A/D modules). It may be possible to power 3.3 V systems with 3.7-4.2 V LiPo batteries. 3.3 V boards will work because the calibration for the sound level meter, multiplying the output voltage by 50 to get the dBA value, means that an output of as much as 3.0 V would produce a value of 150 dBA – a value much larger than any likely ever to be encountered in routine environmental monitoring situations.

To aid in the interpretation of these data, it's at least initially a good idea to provide some personal observations of the sources, timing, and perception of the noise environment so those events can be associated with recorded noise and particulate data.